**Siete stati incaricati di scrivere usando i socket un server a cui possono collegarsi dei citofoni. I citofoni sono collegati al server tramite Ethernet 10Mbps. Il programma server riceve sulla porta 8888 richieste di connessione dai citofoni. Ogni citofono apre due connessioni: una di controllo e l'altra su cui passano i pacchetti dati contenenti la voce campionata. Il server conclude la connessione che trasporta la conversazione quando riceve sulla connessione di controllo un pacchetto contenente la stringa STOP. Fornite lo schema e lo pseudocodice delle principali chiamate socket necessarie per il corretto funzionamento del programma. Gestite correttamente il problema della possibile simultaneità delle richieste di connessione.**

---

## [PUNTO 1]

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

int main ()
{
	int sockid; //descrittore del socket
	struct sockaddr_in serv_addr; //informazioni del server
	int status;

	//inizializzo la struct serv_addr coi parametri definiti nel testo
	serv_addr.sin_addr = localaddress; // assegnamento molto pseudo
	//assegno la porta
	serv_addr.sin_port = 8888;
	//specifico la famiglia
	//per brevità passo la stringa per assegnamento
	serv_addr.sin_family = “AF_INET”;

	//creo il socket e controllo che l'operazione vada a buon fine
	if ((sockid = socket(AF_INET, SOCK_STREAM, 0)) != 0)
	{
		printf(“Errore nella creazione del socket\n”);
		exit(-1);
	}

	//eseguo la bind e verifico che l'operazione vada a buon fine
	if(status_bind = bind (sockid, &serv_addr, sizeof(serv_addr)) != 0)
	{
		print(“Errore nell'esecuzione della bind\n”);
		exit(-1);
	}
```

```
//metto in  ascolto il server e imposto come massima coda di citofoni 10
if(status_listen = listen (sockid, 10) != 0)
{
	print("Errore nell'esecuzione della bind\n");
	exit(-1);
}

//ciclo infinito
for (;;)
{
	struct sockaddr_in client_addr; // informazioni del client chiamante
	int clientid; //socket generato dalla accept
	int pid; //identificativo dei processi padre / figlio

	//eseguo una accept su sockid, così...
	clientid = accept (sockid, &client_addr, sizeof(clientaddr));
	//...ottengo una connessione sul nuovo socket clientid

	//creo un processo figlio
	pid = fork();
	if (pid !=0) //se è il padre...
	{
		close (clientid); //chiudo la connessione tra server e client
	}
	else //se è il figlio...
	{
		int datasocket; //descrittore del nuovo socket per i dati
		int arraySocket[2]; //array di socket, necessari alla select
		boolean fine = FALSE; //si spiegherà nel codice

		//creo il nuovo socket per i dati
		datasocket = socket(AF_INET, SOCK_STREAM, 0);
		//...e controllo che la creazione sia andata a buon fine

		//inizializzo una struct data_addr con le info del nuovo socket
		data_addr.sin_addr = localaddress; //l'IP rimane identico
		//assegno la porta, in modo che sia sempre diversa
		data_addr.sin_port = 8888 + datasocket;
		//specifico la famiglia
		data_addr.sin_family = "AF_INET";

		//eseguo la bind del data_addr
		bind (datasocket, &data_addr, sizeof(data_addr));
		//...e controllo che la bind sia andata a buon fine

		//definisco l'array di socket che verrà scansionato dalla select
		arraySocket[0] = clientid; //...il canale per i comandi
		arraySocket[1] = datasocket; //...il canale per i dati

		//finché la variabile booleana fine non diventa vera...
		while(!fine)
		{
```

```
                        //...effettuo una select sui due socket dell'array
                        int chi = select(2, &arraySocket, NULL, NULL, NULL);
                        //quando sto comunicando col canale dei comandi...
                        if (chi = clientid)
                        {
                                //definisco una stringa per ricevere i comandi
                                string comando;

                                //effettua una receive sul clientid
                                recv(clientid, &comando, sizeof(comando), NULL);
                                //...se ricevo il comando "STOP"...
                                if(comando == "Stop") fine = TRUE;
                                //...allora fine = TRUE
                                //in questo modo potrò uscire dal ciclo while
                        }
                        else //analizza i dati in ingresso
                }
                //il figlio non ha più ragione d'essere, quindi chiudo tutto
                close(clientid);
                close(datasocket);
        }       //...e tanti saluti al figlio
    }  //la fine metafisica del ciclo infinito

    close(sockid); //chiudo tutto...
    return 0; //...e bella lì!
}
```